Adapting to the Shifting Intent of Search Queries[*]

Umar Syed[†]
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
usyed@cis.upenn.edu

Aleksandrs Slivkins
Microsoft Research
Mountain View, CA 94043
slivkins@microsoft.com

Nina Mishra
Microsoft Research
Mountain View, CA 94043
ninam@microsoft.com

Abstract

Search engines today present results that are often oblivious to abrupt shifts in intent. For example, the query 'independence day' usually refers to a US holiday, but the intent of this query abruptly changed during the release of a major film by that name. While no studies exactly quantify the magnitude of intent-shifting traffic, studies suggest that news events, seasonal topics, pop culture, etc account for 50% of all search queries. This paper shows that the signals a search engine receives can be used to both determine that a shift in intent has happened, as well as find a result that is now more relevant. We present a meta-algorithm that marries a classifier with a bandit algorithm to achieve regret that depends logarithmically on the number of query impressions, under certain assumptions. We provide strong evidence that this regret is close to the best achievable. Finally, via a series of experiments, we demonstrate that our algorithm outperforms prior approaches, particularly as the amount of intent-shifting traffic increases.

1 Introduction

Search engines typically use a ranking function to order results. The function scores a document by the extent to which it matches the query, and documents are ordered according to this score. Usually, this function is fixed in the sense that it does not change from one query to another and also does not change over time.

Intuitively, a query is "intent-shifting" if the most desired search result(s) change over time. More concretely, a query's intent has shifted if the click distribution over search results at some time differs from the click distribution at a later time. For the query 'tomato' on the heels of a tomato salmonella outbreak, the probability a user clicks on a news story describing the outbreak increases while the probability a user clicks on the Wikipedia entry for tomatoes rapidly decreases. There are studies that suggest that queries likely to be intent-shifting — such as pop culture, news events, trends, and seasonal topics queries — constitute roughly half of the search queries that a search engine receives [10].

The goal of this paper is to devise an algorithm that quickly adapts search results to shifts in user intent. Ideally, for every query and every point in time, we would like to display the search result that users are most likely to click. Since traditional ranking features like PageRank [4] change slowly over time, and may be misleading if user intent has shifted very recently, we want to use just the observed click behavior of users to decide which search results to display.

There are many signals a search engine can use to detect when the intent of a query shifts. Query features such as as volume, abandonment rate, reformulation rate, occurrence in news articles, and

[*]This is the full version of a paper in *NIPS 2009*.

[†]This work was done while the author was an intern at Microsoft Research and a student in the Department of Computer Science, Princeton University.

the age of matching documents can all be used to build a classifier which, given a query, determines whether the intent has shifted. We refer to these features as the *context*, and an occassion when a shift in intent occurs as an *event*.

One major challenge in building an event classifier is obtaining training data. For most query and date combinations (e.g. 'tomato, 06/09/2008'), it will be difficult even for a human labeler to recall in hindsight whether an event related to the query occurred on that date. In this paper, we propose a novel solution that learns from unlabeled contexts and user click activity.

Contributions. We describe a new algorithm that leverages the information contained in contexts, provided that such information is sufficiently rich. Specifically, we assume that there exists a deterministic oracle (unknown to the algorithm) which inputs the context and outputs a correct binary prediction of whether an event has occurred in the current round. To simulate such an oracle, we use a classification algorithm. However, we do *not* assume that we have a priori labeled samples to train such a classifier. Instead, we generate the labels ourselves.

Our algorithm is in fact a meta-algorithm that combines a bandit algorithm designed for the event-free setting with an online classification algorithm. The classifier uses the contexts to predict when events occur, and the bandit algorithm "starts over" on positive predictions. The bandit algorithm provides feedback to the classifier by checking, soon after each of the classifier's positive predictions, whether the optimal search result actually changed. In such a setup, one needs to overcome several technical hurdles, e.g. ensure that the feedback is not "contaminated" by events in the past and in the near future. We design the whole triad — the bandit algorithm, the classifier, and the meta-algorithm — so as to obtain strong provable guarantees. Our bandit subroutine — a novel version of algorithm UCB1 from [2] which additionally provides high-confidence estimates on the suboptimality of arms — may be of independent interest.

For suitable choices of the bandit and classifier subroutines, the regret incurred by our meta-algorithm is (under certain mild assumptions) at most $O(k + d_{\mathcal{F}})(\frac{n}{\Delta} \log T)$, where k is the number of events, $d_{\mathcal{F}}$ is a certain measure of the complexity of the concept class $\mathcal{F}$ used by the classifier, n is the number of relevant search results,[1] Δ is the "minimum suboptimality" of any search result (defined formally in Section 3), and T is the total number of impressions. This regret bound has a very weak dependence on T, which is highly desirable for search engines that receive much traffic.

The context turns out to be crucial for achieving logarithmic dependence on T. Indeed, we show that any bandit algorithm that ignores context suffers regret $\Omega(\sqrt{T})$, even when there is only one event. Unlike many lower bounds for bandit problems, our lower bound holds even when Δ is a constant independent of T. We also show that assuming a logarithmic dependence on T, the dependence on k and $d_{\mathcal{F}}$ is essentially optimal.

For empirical evaluation, we ideally need access to the traffic of a real search engine so that search results can be adapted based on real-time click activity. Since we did not have access to live traffic, we instead conduct a series of synthetic experiments. The experiments show that if there are no events then the well-studied UCB1 algorithm [2] performs the best. However, when many different queries experience events, the performance of our algorithm significantly outperforms prior methods.

2 Related Work

While there has been a substantial amount of work on ranking algorithms [11, 5, 13, 8, 6], all of these results assume that there is a fixed ranking function to learn, not one that shifts over time. Online bandit algorithms (see [7] for background) have been considered in the context of ranking. For instance, Radlinski et al [20] showed how to compose several instantiations of a bandit algorithm to produce a ranked list of search results. Pandey et al [19] showed that bandit algorithms can be effective in serving advertisements to search engine users. These approaches also assume a stationary inference problem.

Although no existing bandit algorithms are specifically designed for our problem setting, there are two well-known algorithms that we compare against in this paper. The UCB1 algorithm [2] assumes fixed click probabilities and has regret at most $O(\frac{n}{\Delta} \log T)$. The EXP3.S algorithm [3] assumes

[1]In practice, the arms can be restricted to, say, the top ten results that match the query.

that click probabilities can change on every round and has regret at most $O(k\sqrt{nT\log(nT)})$ for arbitrary p_t's. Note that the dependence of EXP3.S on T is substantially stronger.

The "contextual bandits" problem setting [22, 18, 12, 17, 14] is similar to ours. A key difference is that the context received in each round is assumed to contain information about the *identity* of an optimal result i_t^*, a considerably stronger assumption than we make. Our context includes only side information such as volume of the query, but we never actually receive information about the identity of the optimal result.

A different approach is to build a statistical model of user click behavior. This approach has been applied to the problem of serving news articles on the web. Diaz [9] used a regularized logistic model to determine when to surface news results for a query. Agarwal et al [1] used several models, including a dynamic linear growth curve model.

There has also been work on detecting bursts in data streams. For example, Kleinberg [15] describes a state-based model for inferring stages of burstiness. The goal of our work is not to detect bursts, but rather to predict shifts in intent.

In a recent concurrent and independent work, Yu et al [23] studied bandit problems with "piecewise-stationary" distributions, a notion that closely resembles our definition of events. However, they make different assumptions than we do about the information a bandit algorithm can observe. Expressed in the language of our problem setting, they assume that from time-to-time a bandit algorithm receives information about how users *would have* responded to search results that are never actually displayed. For our setting, this assumption is clearly inappropriate.

3 Problem Formulation and Preliminaries

We view the problem of deciding which search results to display in response to user click behavior as a *bandit problem*, a well-known type of sequential decision problem. For a given query q, the task is to determine, at each round $t \in \{1, \ldots, T\}$ that q is issued by a user to our search engine, a single result $i_t \in \{1, \ldots, n\}$ to display.[2] This result is clicked by the user with probability $p_t(i_t)$. A bandit algorithm $\mathcal{A}$ chooses i_t using only observed information from previous rounds, i.e., all previously displayed results and received clicks. The performance of an algorithm $\mathcal{A}$ is measured by its *regret*: $R_{\mathcal{A}}(T) \triangleq E\left[\sum_{t=1}^{T} p_t(i_t^*) - p_t(i_t)\right]$, where an *optimal* result $i_t^* \in \arg\max_i p_t(i)$ is one with maximum click probability, and the expectation is taken over the randomness in the clicks and the internal randomization of the algorithm. Note our unusually strong definition of regret: we are competing against the best result on *every* round.

We call an *event* any round t where $p_{t-1} \neq p_t$. It is reasonable to assume that the number of events $k \ll T$, since we believe that abrupt shifts in user intent are relatively rare. Most existing bandit algorithms make no attempt to predict when events will occur, and consequently suffer regret $\Omega(\sqrt{T})$. On the other hand, a typical search engine receives many signals that can be used to predict events, such as bursts in query reformulation, average age of retrieved document, etc.

We assume that our bandit algorithm receives a *context* $x_t \in \mathcal{X}$ at each round t, and that there exists a function $f \in \mathcal{F}$, in some known *concept class* $\mathcal{F}$, such that $f(x_t) = +1$ if an event occurs at round t, and $f(x_t) = -1$ otherwise.[3] In other words, f is an *event oracle*. The tractability of $\mathcal{F}$ will be characterized by a number $d_{\mathcal{F}}$ called the *diameter* of $\mathcal{F}$, detailed in Section 5. At each round t, an *eventful bandit algorithm* must choose a result i_t using only observed information from previous rounds, i.e., all previously displayed results and received clicks, plus all contexts up to round t.

In order to develop an efficient eventful bandit algorithm, we make an additional key assumption: At least one optimal result before an event is *significantly* suboptimal after the event. More precisely, we assume there exists a *minimum shift* $\epsilon_S > 0$ such that, whenever an event occurs at round t, we have $p_t(i_{t-1}^*) < p_t(i_t^*) - \epsilon_S$ for at least one previously optimal search result i_{t-1}^*. For our problem setting, this assumption is relatively mild: the events we are interested in tend to have a

[2]For simplicity, we focus on the task of returning a single result, and not a list of results. Techniques from [20] may be adopted to find a good list of results.

[3]In some of our analysis, we require contexts be restricted to a strict (concept-specific) subset of $\mathcal{X}$; the value of f outside this subset will technically be `null`. See Section 5 for more details.

Table 1: Notation

$\mathcal{X}$	universe of contexts	k	number of events
$x_t \in \mathcal{X}$	context in round t	n	number of arms
$\mathcal{F}$	concept class	T	time horizon
$f \in \mathcal{F}$	concept	ϵ_S	min shift of an event
$d_\mathcal{F}$	diameter of $\mathcal{F}$	Δ	min suboptimality of an arm

rather dramatic effect on the optimal search results. Moreover, our bounds are parameterized by $\Delta = \min_t \min_{i \neq i_t^*} p_t(i_t^*) - p_t(i)$, the *minimum suboptimality* of any suboptimal result.

We summarize the notation in Table 1.

Let S be the set of all contexts which correspond to an event. When the classifier receives a context x and predicts a "positive", this prediction is called a *true positive* if $x \in S$, and a *false positive* otherwise. Likewise, when the classifier predicts a "negative", the prediction is called a *true negative* if $x \notin S$, and a *false negative* otherwise. The sample (x, l) is *correctly labeled* if $l = (x \in S)$.

4 Bandit with Classifier

Our algorithm is called BWC, or "Bandit with Classifier". Ideally, we would like to use a bandit algorithm for the event-free setting, such as UCB1, and restart it every time there is an event. Since we do not have an oracle to tell whether an event has happened, we use a classifier which looks at the current context and makes a binary prediction. As we mentioned in the introduction, we assume that a priori there are no labeled samples to train such a classifier, so we need to generate the labels ourselves. The high-level idea is to restart the bandit algorithm every time the classifier predicts an event, and use subsequent rounds to generate feedback (labeled samples) to train the classifier. Thus, we have a *feedback loop* between the bandit algorithm and a classifier, in which the latter provides predictions and the former verifies whether they are correct, see Figure 1.

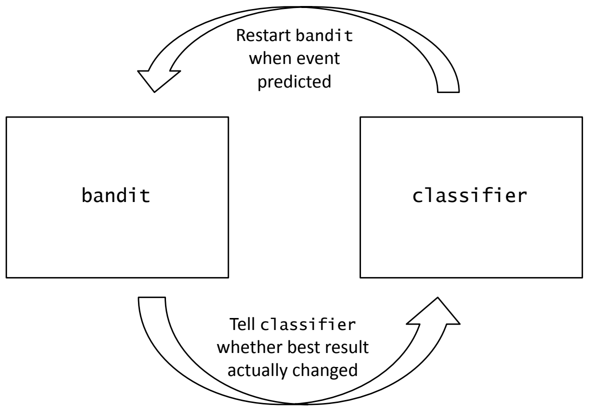

Restart bandit when event predicted
bandit
classifier
Tell classifier whether best result actually changed

Figure 1: A high-level picture of the operation of the BWC algorithm, depicting the feedback loop between the two main subroutines, `bandit` and `classifier`.

So what prevents us from simply combining an off-the-shelf bandit algorithm with an off-the-shelf classifier? The central challenge is how to define the feedback. Let us outline several hurdles that we need to overcome here. A single false negative prediction will cause BWC to miss an event, which may result in a very high regret (since it may take the bandit algorithm a very long time to adjust). Incorrectly labeled samples may contaminate the classifier, perhaps even permanently. To generate a label for a given sample, one needs to compare the state right before the current round with the state right after, in a conclusive way. Both states are not known to the algorithm a priori, and can only be learned probabilistically via exploration. A particular challenge is to ensure that such exploration is not contaminated by events in the past rounds, as well as by events that happen soon after the current round. Moreover, this exploration is generally too expensive to perform upon *negative* predictions — indeed, the whole point of BWC is that in the absence of an event the bandit algorithm converges to the best arm and (essentially) keeps playing it — so the classifier receives labels only upon the positive predictions.

4.1 The meta-algorithm

We will present our algorithm in a modular way, as a *meta-algorithm* which uses the following two components: `classifier` and `bandit`. In each round, `classifier` inputs a context x_t and outputs a "positive" or "negative" prediction of whether an event has happened in this round. Also, it may input labeled samples of the form (x, l), where x is a context and l is a boolean label, which it uses for training. Algorithm `bandit` is a bandit algorithm that is tuned for the event-free runs.

As described above, we further require `bandit` to provide feedback to the classifier about whether the best result has actually changed. The standard bandit framework does not immediately provide us with estimates from which such feedback can be obtained. Therefore we require `bandit` to provide the following additional functionality: after each round t of execution, it outputs a pair (G^+, G^-) of subsets of arms;[4] we call this pair the *t-th round guess*.[5] The meaning of G^+ and G^- is that they are algorithm's estimates for, respectively, the sets of all optimal and (at least) ϵ_S-suboptimal arms. We use (G^+, G^-) to predict whether an event has happened between two runs of `bandit`. The idea is that any such event causes some arm from G^+ of the first run to migrate to G^- of the second run. Accordingly, we generate a negative label if $G_i^+ \cap G_j^- = \emptyset$, where i and j refers to the first and the second run, respectively (see Line 10 of Algorithm 1).

We formalize our assumptions on `classifier` and `bandit` as follows:

Definition 1. `classifier` *is* **safe** *for a given concept class if, given only correctly labeled samples, it never outputs a false negative.* `bandit` *is called (L, ϵ)-***testable***, for some $L \in \mathbb{N}$ and $\epsilon \in (0, 1)$, if the following holds. Consider an event-free run of* `bandit`*, and let (G^+, G^-) be its L-th round guess. Then with probability at least $1 - T^{-2}$, each optimal arm lies in G^+ but not in G^-, and any arm that is at least ϵ-suboptimal lies in G^- but not in G^+.* [6]

We will discuss efficient implementations of a safe `classifier` and a (L, ϵ_S)-testable `bandit` in Sections 5 and Section 6, respectively. For `bandit`, we build on a standard algorithm UCB1 [2]; as it turns out, making it (L, ϵ_S)-testable requires a significantly extended analysis.

For correctness, we require `bandit` to be (L, ϵ_S)-testable, where ϵ_S is the minimum shift. The performance of `bandit` is quantified via its **event-free regret**, i.e. regret on the event-free runs. Likewise, for correctness we need `classifier` to be safe, and we quantify its performance via the following property, termed **FP-complexity**, which refers to the maximum number of false positives.

Definition 2. *Given a concept class $\mathcal{F}$, the FP-complexity of* `classifier` *is the maximum possible number of false positives it can make in an online prediction game where in each round, an adversary selects a sample,* `classifier` *makes a prediction, and then (in some rounds) receives a correct label. Specifically,* `classifier` *receives a correct label if and only if the prediction is a false positive. The maximum is taken over all event oracles $f \in \mathcal{F}$ and all possible sequences of samples.*

Now we are ready to present our meta-algorithm, called BWC. It runs in phases of two alternating types: odd phases are called "testing" phases, and even phases are called "adapting" phases. The first round of phase j is denoted t_j. In each phase we run a fresh instance of `bandit`. Each testing phase lasts for L rounds, where L is a parameter. Each adapting phase j ends as soon as `classifier` predicts "positive"; the round t when this happens is round t_{j+1}. Phase j is called *full* if it lasts at least L rounds. For a full phase j, let (G_j^+, G_j^-) be the L-th round guess in this phase. After each testing phase j, we generate a boolean prediction l of whether there was an event in the first round thereof. Specifically, letting i be the most recent full phase before phase j, we set $l_{t_j} =$ `false` if and only if $G_i^+ \cap G_j^- = \emptyset$. If l_{t_j} is `false`, the labeled sample (x_{t_j}, l_{t_j}) is fed back to the classifier.

[4]Following established convention, we call the options available to a bandit algorithm "arms". In our setting, each arm corresponds to a search result.

[5]Both `classifier` and `bandit` make predictions (about events and arms, respectively). For clarity, we will use the term "guess" exclusively to refer to predictions made by `bandit`, and reserve the term "prediction" for `classifier`.

[6]Recall that T here is the overall time horizon, as defined in Section 3.

Algorithm 1 Meta-algorithm BWC (“Bandit with Classifier”)

```
1: Given: Parameter $L$, a $(L, \epsilon_S)$-testable bandit, and a safe classifier.
2: for phase $j = 1, 2, \ldots$ do
3:     Initialize bandit. Let $t_j$ be current round.
4:     if $j$ is odd then
5:         {testing phase}
6:         for round $t = t_j \ \ldots \ t_j + L$ do
7:             Select arm $i_t$ according to bandit.
8:             Observe click w.p. $p_t(i_t)$ and update bandit.
9:         Let $i$ be the most recent full phase before phase $j$.
10:        If $G_i^+ \cap G_j^- = \emptyset$      {label is false}
11:            Let $l_{t_j}$ = false and pass training example $(x_{t_j}, l_{t_j})$ to classifier.
12:    else
13:        {adapting phase}
14:        for round $t = t_j,\ t_j + 1, \ldots$ do
15:            Select arm $i_t$ according to bandit.
16:            Observe click w.p. $p_t(i_t)$ and update bandit; pass context $x_t$ to classifier.
17:            if classifier predicts “positive” then
18:                Terminate inner for loop.
```

Note that `classifier` never receives `true`-labeled samples. Pseudocode for BWC is given in Algorithm 1.

Disregarding the interleaved testing phases for the moment, BWC restarts `bandit` whenever `classifier` predicts “positive”, optimistically assuming that the prediction is correct. By our assumption that events cause some optimal arm to become significantly suboptimal (see Section 3), a correct prediction should result in $G_i^+ \cap G_j^- \neq \emptyset$, where i is a phase before the putative event, and j is a phase after it. We use this condition in Line 10 of the pseudocode to generate the label. However, to ensure that the estimates G_i and G_j are reliable, we require that phases i and j are full. And to ensure that the full phases closest to a putative event are not too far from it, we interleave a full testing phase every other phase.

4.2 Provable guarantees

We present provable guarantees for BWC in a modular way, in terms of FP-complexity, event-free regret, and the number of events. This is the main technical result in the paper.

Theorem 1. *Consider an instance of the eventful bandit problem with number of rounds T, n arms, k events and minimum shift ϵ_S; assume that any two events are at least $2L$ rounds apart. Consider algorithm* BWC *with parameter L and components* `classifier` *and* `bandit` *that are, respectively, safe and (L, ϵ_S)-testable. Suppose the event-free regret of* `bandit` *is bounded from above by a concave function $R_0(\cdot)$. Then the regret of* BWC *is*

$$R_{\text{BWC}}(T) \leq (2k + d_{\text{FP}})\, R_0\left(\tfrac{T}{2k+d_{\text{FP}}}\right) + (k + d_{\text{FP}})\, R_0(L) + kL, \tag{1}$$

where d_{FP} is the FP-complexity of `classifier`.

Remark. We define a safe classifier whose FP-complexity is bounded in terms of some properties of the underlying concept class (see Section 5). Our instantiation of `bandit` is (L, ϵ_S)-testable for $L = \Theta(\frac{n}{\epsilon_S^2} \log T)$, with concave event-free regret matching that of UCB1 (see Section 6).

Remark. The right-hand side of (1) can be parsed as follows. The three summands in (1) correspond to contributions of, respectively, adapting phases, event-free testing phases, and testing phases during which an event has occurred. For the first summand, we show that BWC incurs regret $R_0(t)$ for each adapting phase of length t, bound the number of adapting phases by $2k + d_{\text{FP}}$, and then bound the total contribution of all such phases using concavity. For the second summand, we bound the number of clean testing phases by $k + d_{\text{FP}}$, and note that each such phase contributes at most $R_0(L)$

to regret. For the third summand, each "eventful" testing phase contributes at most L to regret, and we show that there can be at most k such phases.[7]

Remark. Assuming that any two events are at least $2L$ rounds apart ensures that of any two consecutive phases, one much be event-free. This, in turn, let us invoke the (L, ϵ_S)-testability of `bandit`.

Overview of the proof. The essential difficulty the analysis of BWC is that an event might happen while the algorithm is testing for another (suspected) event. The corresponding technical difficulty is that the correct operation of the components of BWC — `classifier` and `bandit` — is interdependent, so one needs to be careful to avoid a circular argument. In particular, one challenge is to handle events that occur during the first L rounds of a phase; these events may potentially "contaminate" the L-th round guesses and cause incorrect feedback to `classifier`.

First, we argue away the probabilistic nature of the problem. We focus on a given testing phase j. For each of the two preceding phases $i \in \{j-1, j-2\}$, consider the number N of events between the first round of phase i and the first round of phase j. We would like to establish the following *separation property*: that we can separate (tell apart) the cases of $N = 0$ and $N = 1$ using the testing condition in Line 10 of the pseudocode. Capitalizing on (L, ϵ_S)-testability, we define a technical condition which implies the separation property with very high probability. Regret incurred if the implication "technical condition $\Rightarrow$ separation property" [8] fails to hold is negligible. Thus we can assume that this implication holds always, and argue deterministically from now on.

It is worth noting that we consider *two* preceding phases $i \in \{j-1, j-2\}$ because either can be used in in Line 10 of the pseudocode (depending on whether phase $j-1$ is full). A crucial point here is that one of these two phases must be event-free.

Second, we argue that `classifier` receives only correctly labeled samples. We do it in two steps. Using the well-detectable property, we show that if `classifier` receives an incorrectly labeled sample after some testing phase j, then an event must have occurred during the (adapting) phase $j-1$. Then using the safety property of `classifier`, we prove that each adapting phase is event-free.

Third, we bound from above the number of testing and adapting phases, using the maximal number of events and the FP-complexity of the classifier. To this end, we establish that if during a testing phase j there are no events, and furthermore there are no events during the two preceding phases, then in the end of j BWC generates a correct label $l = \texttt{false}$. Then the regret bound (1) follows easily from the event-free regret of `bandit`. □

Now let us present the full proof which fills the gaps in the above overview.

Full Proof. Let t_j be the first round of phase j. Recall that phase j is called *full* if it lasts at least L rounds. For a full phase j, let us say that the phase is *event-free* if no events happened during interval $(t_j,\, t_j + L]$, and let (G_j^+, G_j^-) be the L-th round guess in this phase. For two full phases $i < j$, let us write $i \oplus j$ if and only if $G_i^+ \cap G_j^- = \emptyset$. Recall that $i \oplus j$ (as a boolean property) is our algorithm's estimate of whether there was no event in round t_j.

A testing phase j is called *well-detectable* if for each phase $i \in \{j-2, j-1\}$ the following property holds: if phases i and j are full and event-free, then: (i) if there are no events in the interval $(t_i,\, t_j]$ then $i \oplus j$, (ii) if in the interval $(t_i,\, t_j]$ there is exactly one event, then $\neg(i \oplus j)$. Since `bandit` is (L, ϵ_S)-testable, each testing phase j is well-detectable with probability at least $1 - 2T^{-2}$. Thus, with probability at least $1 - \Omega(T^{-1})$ each testing phase is well-detectable. Thus, regret incurred in the case that a phase fails to be well-detectable is negligible. So in the rest of the proof, we will assume that each testing phase is well-detectable.

We claim that if `classifier` receives an incorrectly labeled sample after some testing phase j, then an event must have occurred during the (adapting) phase $j-1$. Indeed, by the algorithm specification this sample is $(x_{t_j}, \texttt{false})$, where t_j is the first round of phase j. Thus, an event has happened in round t_j, and yet we have $i \oplus j$, where i is the most recent full phase before phase j.

[7]In fact, the k in the $+kL$ term in (1) can be replaced by the (potentially much smaller) number of testing phases that contain both a false positive in round 1 of the phase and an actual event later in the phase.

[8]In the full proof, this implication is called *well-detectability*.

Since each testing phase is well-detectable, it follows that at least one more event happened between the beginning of phase i and the end of phase j. Since any two events are at least $2L$ rounds apart, phase i started at some round $t_i < t_j - 2L$, and an event has happened in the interval $[t_i,\, t_j - 2L)$. To prove the claim, it suffices to show that $i = j - 1$. Now, if phase $j - 1$ lasted less than L steps, then $i = j - 2$ is a testing phase, and so $t_i \geq t_j - 2L$, contradiction. Thus phase $j - 1$ lasted at least L steps, and so $i = j - 1$, claim proved.

We claim that all adapting phases are event-free. For the sake of contradiction, suppose an event occurs during an adapting phase, and let t be the first round at which this happens. We know that `classifier` output a (false) negative in this round, since otherwise a new testing phase would have started at round t. Since `classifier` is safe, at some round before t it must have received an incorrectly labeled sample. By the algorithm specification, this must have happened after some testing phase j which ended before round t. But then (by the previous claim) an event must have occurred during the (adapting) phase $j - 1$, which contradicts the choice of t. Claim proved.

From the previous two claims, it follows that `classifier` receives only correctly labeled samples.

We claim that if there are no events during some testing phases $j - 2$ and j, then at the end of phase j we generate a label $l = \texttt{false}$. Indeed, suppose not. Then $\neg(i \oplus j)$, where i is the most recent full phase before phase j. Either $i = j - 2$ or $i = j - 1$; in either case, $\neg(i \oplus j)$ implies that there is an event in the interval $[t_i, t_j)$. Since there are no events during adapting phases, it follows that $i = j - 2$, contradiction. Claim proved.

We claim that there can be at most $2k + d_{\mathrm{FP}}$ testing phases (and hence at most as many adapting phases), including at most $k + d_{\mathrm{FP}}$ event-free testing phases. Indeed, in the first round of each testing phase j `classifier` generates a "positive", and in the end of the phase we generate a label $l \in \{\texttt{true}, \texttt{false}\}$. We examine each case separately: (i) if $l = \texttt{false}$ then `classifier` receives feedback, so there can be at most d_{FP} such phases j, (ii) if $l = \texttt{true}$ then an event has occurred in phase j or $j - 2$, so there can be at most $2k$ such phases j, of which at most k phases can be event-free. Claim proved.

To obtain the regret bound (1), note that regret in each event-free phase of length t is $R_0(t)$, see the second remark after Theorem 1 for details. □

5 Safe Classifier

In this section, we show how safe classifiers with low FP-complexity can be constructed for specific concept classes. Recall that a classifier is called safe if (assuming it inputs only correctly labeled samples) it never outputs a false negative, and the definition of FP-complexity, motivated by the specification of the BWC algorithm, essentially assumes that all labeled samples correspond to false positives.

We first describe a generic classifier, called `SafeCl`, that is safe for *any* concept class $\mathcal{F}$, and bound its FP-complexity using a certain property of $\mathcal{F}$. In the event that the concept class is all d-dimensional axis-parallel hyper-rectangles with margin $1/\delta$, we show that this bound is proportional to d/δ. And in the event that the concept class is all d-dimensional hyperplanes with margin δ, we show that this bound is exponential in d. Unfortunately, the exponential dependence cannot be improved, as we will see in Section 7.

The classifier `SafeCl` is defined as follows.

> `SafeCl` classifies a given unlabeled context x as negative if and only if there exists no concept $f \in \mathcal{F}$ such that $f(x) = +1$ and $f(x') = -1$ for each `false`-labeled example x' received so far.

It is easy to see that this classifier is indeed safe. Moreover, we bound its FP-complexity in terms of the following property of the concept class $\mathcal{F}$:

Definition 3. *The* diameter *of $\mathcal{F}$, denoted $d_{\mathcal{F}}$, is equal to the length of the longest sequence $x_1, \ldots, x_m \in \mathcal{X}$ such that for each $t = 1, \ldots, m$ there exists a concept $f \in \mathcal{F}$ with the following property: $f(x_t) = +1$, and $f(x_s) = -1$ for all $s < t$.*

Claim 1. `SafeCl` *is safe, and its FP-complexity is at most $d_{\mathcal{F}}$.*

Proof. Assume all `false`-labeled examples input by `SafeCl` are correctly labeled. Suppose `SafeCl` outputs a false negative, with concept $f \in \mathcal{F}$ and unlabeled sample x. Then $f(x) = +1$ and $f(x') = -1$ for each `false`-labeled example x' received so far. But by definition of `SafeCl` such concept does not exist, contradiction. Therefore, `SafeCl` is safe. Regarding the FP-complexity, consider the prediction game in Definition 2. Any sequence $x_1, \ldots, x_m$ of false positives output by `SafeCl` satisfies the property in Definition 3, so $m \leq d_{\mathcal{F}}$. □

By using `SafeCl` as our classifier, we introduce $d_{\mathcal{F}}$ into the regret bound of `bwc`, and this quantity can be large. However, in Section 7 we show that the regret of *any* algorithm must depend on $d_{\mathcal{F}}$, unless it depends strongly on the number of rounds T.

Below we give examples of common concept classes with efficiently computable safe functions, and prove bounds on their diameter. Recall that for a given universe $\mathcal{X}$ of examples, a concept is a function $f : \mathcal{X} \to \{-1, +1, \texttt{null}\}$, where the `null` value refers to the examples that are not feasible under a given concept (i.e., if f is the true concept, then we will never observe an example x such that $f(x) = \texttt{null}$).

In what follows, for each $N \subset \mathcal{X}$ define $S_{\mathcal{F}}(N) \subset \mathcal{X}$ as the set of all $x \in \mathcal{X}$ for which there is no concept $f \in \mathcal{F}$ such that $f(x) = +1$ and $f(x') = -1$ for each $x' \in N$. Note that `SafeCl` outputs a negative prediction on x if and only if $x \in S_{\mathcal{F}}(N)$, where N is the set of `false`-labeled samples received so far. Likewise, in Definition 3 the sequence $\{x_t\}$ satisfies $x_t \notin S_{\mathcal{F}}(\{x_1, \ldots, x_{t-1}\})$ for each t.

For convenience, define a "δ-ball" around a set $S \subset \mathbb{R}^d$ in the d-dimensional L_p-norm as

$$\mathbb{B}_p^d(S, \delta) \triangleq \{x \in \mathbb{R}^d : \; L_p(x, S) \leq \delta\}, \text{ where } L_p(x, S) \triangleq \min_{y \in S} \|x - y\|_p.$$

Here $L_p(x, S)$ is the L_p-norm distance between a point x and a set S.

5.1 Axis-parallel rectangles with margin δ

One very simple concept is an axis-parallel hyper-rectangle. This type of concept can be used to test whether any one of several features is outside of its 'normal' range. This is a particularly well-suited concept class for predicting events that may affect a search engine query, since these events are typically preceded by a large change in some statistic related to the query, such as its volume or abandonment rate.

Fix the dimension d, and let $\mathcal{X} \subseteq \mathbb{R}^d$ be the d-dimensional L_∞-norm unit ball around the origin. A d-*rectangle* in $\mathbb{R}^d$ is the cross-product of d non-empty intervals in $\mathbb{R}$. Given $\delta > 0$ and a d-rectangle R, define a function $f_{R,\delta} : \mathcal{X} \to \{-1, +1, \texttt{null}\}$ as follows: $f_{R,\delta}(x)$ equals $+1$ if $L_\infty(x, R) \geq \delta$; it equals -1 if $x \in R$, and it equals `null` otherwise (note that the margin δ only applies only outside of R). The concept class of d-dimensional *axis-parallel rectangles with margin* δ is defined as $\mathcal{F}_{\texttt{APR}(d,\,\delta)} = \{f_{R,\delta} : \text{ all } d\text{-rectangles } R\}$.

We bound the diameter of $\mathcal{F}_{\texttt{APR}(d,\,\delta)}$ as follows.

Claim 2. *If $\mathcal{F} = \mathcal{F}_{\texttt{APR}(d,\,\delta)}$, then $d_{\mathcal{F}} \leq O(d/\delta)$.*

Proof. Consider a sequence $x_1, \ldots, x_m \in \mathcal{X}$ such that $x_t \notin S_{\mathcal{F}}(\{x_1, \ldots, x_{t-1}\})$ for all $1 \leq t \leq m$. Let R_t be the δ-ball in L_∞ around the smallest d-rectangle containing $x_1, \ldots, x_t$. By definition of the sequence, at least one of the one-dimensional intervals defining R_{t+1} must be δ larger than the same interval in R_t. Since $\|x_t\|_\infty \leq 1$, $m \leq O(d/\delta)$. □

Clearly, for the concept class $\mathcal{F}_{\texttt{APR}(d,\,\delta)}$, the classifier `SafeCl` simply maintains the smallest d-dimensional rectangle $R(N)$ containing the set of all previously `false`-labeled examples N, and classifies a new example x as negative if and only if x lies within δ (measured in L_∞-norm) of $R(N)$. In other words

`SafeCl` on $\mathcal{F}_{\texttt{APR}(d,\,\delta)}$: classify $x \in \mathcal{X}$ as negative $\iff$ $x \in \mathbb{B}_\infty^d(R(N),\, \delta)$, where N is the set of all `false`-labeled examples received so far.

5.2 Hyperplanes with margin δ

Hyperplanes are perhaps the most widely-used concept in classification problems. Fix the dimension d, and let $\mathcal{X} \subseteq \mathbb{R}^d$ be the d-dimensional L_2-norm unit ball around the origin. Given $u, w \in \mathbb{R}^d$ and $\delta > 0$, define a function $f_{u,w,\delta} : \mathcal{X} \to \{-1, +1, \mathtt{null}\}$ as follows: $f_{u,w,\delta}(x)$ equals $+1$ if $w \cdot (x+u) \geq \delta$, it equals -1 if $w \cdot (x+u) < -\delta$, and it equals `null` otherwise. Here w is the unit normal of the hyperplane, and u is the shift vector. The concept class of d-dimensional *hyperplanes with margin* δ is defined as

$$\mathcal{F}_{\mathsf{HYP}(d,\,\delta)} = \{f_{u,w,\delta} : \; u, w \in \mathbb{R}^d, \; \|u\|_2 \leq 1, \|w\|_2 = 1\}.$$

We bound the diameter of $\mathcal{F}_{\mathsf{HYP}(d,\,\delta)}$ as follows:

Claim 3. *If $\mathcal{F} = \mathcal{F}_{\mathsf{HYP}(d,\,\delta)}$, then $d_{\mathcal{F}} \leq (1 + \frac{1}{\delta})^d$.*

Proof. Consider a sequence $x_1, \ldots, x_m \in \mathcal{X}$ such that $x_t \notin S_{\mathcal{F}}(\{x_1, \ldots, x_{t-1}\})$ for all $1 \leq t \leq m$, as in Definition 3. Then for each s and t such that $1 \leq s < t \leq m$ there exist $u, w \in \mathbb{R}^d$ such that $\|u\|_2 \leq 1$, $\|w\|_2 = 1$, $w \cdot (x_t + u) \geq \delta$ and $w \cdot (x_s + u) < -\delta$. By Hölder's inequality, it follows that

$$\|x_t - x_s\|_2 = \|w\|_2 \, \|x_t - x_s\|_2 \geq w \cdot (x_t - x_s) > 2\delta. \tag{2}$$

Now, place an L_2-ball of radius δ around each point x_t. By (2), none of these balls can intersect. A radius-r ball in d dimensions has volume $C_d \, r^d$, where C_d is a constant that depends only on d. Thus the total volume of the balls is $m \, C_d \, \delta^d$. On the other hand, $\|x_t\|_2 \leq 1$ for each t, so each of these balls lies in the radius-$(1+\delta)$ ball around the origin, so their total volume is at most $C_d \, (1+\delta)^d$. It follows that $m \leq (1 + \frac{1}{\delta})^d$. □

We now show that there is a computationally efficient way to implement the classifier `SafeCl` for hypothesis class $\mathcal{F}_{\mathsf{HYP}(d,\,\delta)}$. Specifically, we show that the classifier `SafeCl` simply maintains the convex hull $\mathtt{Co}(N)$ of all previously `false`-labeled examples N, classifies a new example x as negative if and only if x lies within 2δ (measured in L_2-norm) of $\mathtt{Co}(N)$. In other words

`SafeCl` on $\mathcal{F}_{\mathsf{HYP}(d,\,\delta)}$: classify $x \in \mathcal{X}$ as negative $\iff$ $x \in \mathbb{B}_2^d(\mathtt{Co}(N),\, 2\delta)$, where N is the set of all `false`-labeled examples received so far.

Claim 4. *If $\mathcal{F} = \mathcal{F}_{\mathsf{HYP}(d,\,\delta)}$ and $N \subset \mathcal{X}$ then $S_{\mathcal{F}}(N) = \mathcal{X} \cap \mathbb{B}_2^d(\mathtt{Co}(N),\, 2\delta)$, where $\mathtt{Co}(N)$ is the convex hull of N.*

Proof. Fix $x_t \in \mathcal{X}$. We divide the proof into two parts. First, we show that if x_t is contained in the 2δ-ball around $\mathtt{Co}(N)$, then no hyperplane in $\mathcal{F}$ can separate x_t from N. Next, we show that if x_t is outside the 2δ-ball around $\mathtt{Co}(N)$, then at least one hyperplane in $\mathcal{F}$ separates x_t from N. More precisely, we prove that

(i) If $x_t \in \mathbb{B}_2^d(\mathtt{Co}(N),\, 2\delta)$ then there does not exist $f \in \mathcal{F}_{\mathsf{HYP}(d,\,\delta)}$ such that $f(x_s) = -1$ for all $x_s \in N$ and $f(x_t) = +1$.

(ii) If $x_t \notin \mathbb{B}_2^d(\mathtt{Co}(N),\, 2\delta)$ then there exists $f \in \mathcal{F}_{\mathsf{HYP}(d,\,\delta)}$ such that $f(x_s) = -1$ for all $x_s \in N$ and $f(x_t) = +1$.

Proof of (i): Suppose for contradiction that there exist $u, w \in \mathbb{R}^d$, with $\|u\|_2 \leq 1$ and $\|w\|_2 = 1$, such that $w \cdot (x_s + u) < -\delta$ for all $x_s \in N$ and $w \cdot (x_t + u) \geq \delta$.

Choose $x^* \in \mathtt{Co}(N)$ so that $\|x_t - x^*\|_2 = L_2(x_t, \mathtt{Co}(N))$, i.e. x^* is a closest point in $\mathtt{Co}(N)$ to x_t (we know x^* exists because $\mathtt{Co}(N)$ is closed). Since $x_t \in \mathbb{B}_2^d(\mathtt{Co}(N),\, 2\delta)$, we have that $\|x_t - x^*\|_2 \leq 2\delta$.

We know that $w \cdot (x^* + u) < -\delta$ because x^* is a convex combination of the examples in N. Therefore, by the intermediate value theorem, there exists $x' \in \mathcal{X}$ and $\theta \in [0,1]$ such that $x' = (1-\theta)x_t + \theta x^*$ and $w \cdot (x' + u) = 0$.

Some algebra shows that $\|x_t - x'\|_2 = \theta\|x_t - x^*\|_2$ and $\|x' - x^*\|_2 = (1-\theta)\|x_t - x^*\|_2$. Adding these equations yields

$$\|x_t - x'\|_2 + \|x' - x^*\|_2 = \|x_t - x^*\|_2$$

Because $w \cdot (x' + u) = 0$, by Hölder's inequality we have

$$\|x_t - x'\|_2 = \|w\|_2\|x_t - x'\|_2 \geq w \cdot (x_t - x') = w \cdot (x_t + u) - w \cdot (x' + u) \geq \delta$$

and

$$\|x' - x^*\|_2 = \|w\|_2\|x' - x^*\|_2 \geq w \cdot (x' - x^*) = w \cdot (x' + u) - w \cdot (x^* + u) > \delta$$

which implies $\|x_t - x^*\|_2 > 2\delta$, which is a contradiction.

Proof of (ii): We will use the well-known *separating hyperplane theorem* [21]: If nonempty convex sets $X, Y \in \mathbb{R}^d$ do not intersect, then there exist $a \in \mathbb{R}^d \setminus \{0\}$ and $b \in \mathbb{R}$ such that

$$a \cdot x \geq b \text{ for all } x \in X \text{ and } a \cdot y \leq b \text{ for all } y \in Y \tag{3}$$

Since $x_t \notin B_2^d(\texttt{Co}(N), 2\delta)$ there must exist $\epsilon > 0$ such that the sets $X = B_2^d(\{x_t\}, \delta)$ and $Y = B_2^d(\texttt{Co}(N), \delta + \epsilon)$ do not intersect. For these choices for X and Y, let us fix $a \in \mathbb{R}^d \setminus \{0\}$ and $b \in \mathbb{R}$ that satisfy (3).

Note that $x_t + z \in X$ for all $z \in \mathbb{R}^d$ such that $\|z\|_2 \leq \delta$. Also note that $x_s + z \in Y$ for all $x_s \in N$ and $z \in \mathbb{R}^d$ such that $\|z\|_2 \leq \delta + \epsilon$. So by (3) we have

$$a \cdot \left(x_t - \delta\frac{a}{\|a\|_2}\right) \geq b \text{ and } a \cdot \left(x_s + (\delta + \epsilon)\frac{a}{\|a\|_2}\right) \leq b \text{ for all } x_s \in N$$

Letting $w = \frac{a}{\|a\|_2}$ and rearranging we have

$$w \cdot x_t \geq \frac{b}{\|a\|_2} + \delta \text{ and } w \cdot x_s \leq \frac{b}{\|a\|_2} - (\delta + \epsilon) \text{ for all } x_s \in N \tag{4}$$

Since $\|w\|_2 = 1$ and $\|x\|_2 \leq 1$ for all $x \in \mathcal{X}$, it follows from (4) that $\left|\frac{b}{\|a\|_2}\right| \leq 1$. Thus there exists $u \in \mathbb{R}^d$ such that $\|u\|_2 \leq 1$ and $w \cdot u = -\frac{b}{\|a\|_2}$. It now follows that

$$w \cdot (x_t + u) \geq \delta \text{ and } w \cdot (x_s + u) \leq -\delta - \epsilon \text{ for all } x_s \in N$$

So the function $f_{u,w,\delta} \in \mathcal{F}_{\texttt{HYP}(d,\,\delta)}$ satisfies the claim. □

6 Testable Bandit Algorithms

In this section we will consider the stochastic n-armed bandit problem. We are looking for (L, ϵ)-testable algorithms with low regret. The L will need to be sufficiently large, on the order of $\Omega(n\epsilon^{-2})$.

A natural candidate would be algorithm UCB1 from [2] which does very well on event-free regret:

$$R_0(L) \leq O(\min(\tfrac{n}{\Delta}\log L,\ \sqrt{nL\log L})). \tag{5}$$

Unfortunately, UCB1 does not immediately provide a way to define the t-th round best guess (G^+, G^-) so as to guarantee (L, ϵ)-testability. One simple fix is to choose an arm at random in each of the first L rounds, use these samples to form the best guess, in a straightforward way, and then run UCB1. However, in the first L rounds this algorithm incurs regret of $\Omega(L)$, which is very suboptimal compared to $R_0(L)$ from (5).

In this section, we develop an algorithm which has the same regret bound as UCB1, and is (L, ϵ)-testable. We state this result more generally, in terms of estimating expected payoffs; we believe it may be of independent interest. The (L, ϵ)-testability is then an easy corollary.

Since our analysis in this section is for the event-free setting, we can drop the subscript t from much of our notation. Let $p(u)$ denote the (time-invariant) expected payoff of arm u. Let $p^* = \max_u p(u)$, and let $\Delta(u) = p^* - p(u)$ be the "suboptimality" of arm u. For round t, let $\mu_t(u)$ be the sample average of arm u, and let $n_t(u)$ be the number of times arm u has been played.

Algorithm 2 The (L,ϵ)-testable bandit algorithm with low regret.

```
1: Given: Time horizon $T$, parameter $\epsilon \in (0,1)$.
2: for all arms $u$ do
3:     $n(u) \leftarrow 0$, $x(u) \leftarrow 0$, $\mu(u) \leftarrow 0$    {#samples, total reward, sample average}
4: for rounds $t = 1, 2, \ldots, T$ do
5:     Pick arm $u$ with the maximal index $I(u) = \mu(u) + 12\sqrt{\frac{2\,\log(t+T)}{1+n(u)}}$.
6:     Observe payoff $x$, update $n(u) \leftarrow n(u)+1$,  $x(u) \leftarrow x(u)+x$,  $\mu(u) \leftarrow x(u)/n(u)$.
7:     { Form the $t$-th round guess }
8:     $v^* \leftarrow$ arm played most often in the last $t/2$ rounds.
9:     for all arms $v$ do
10:        $\widehat{\Delta}(v) \leftarrow \mu(v^*) - \mu(v)$      {the $t$-th round estimate of $\Delta(v)$}
11:    Output $(G^+, G^-) = \Big(\{v : \widehat{\Delta}(v) \leq \epsilon/4\},\ \{v : \widehat{\Delta}(v) > \epsilon/2\}\Big)$.
```

We will use a slightly modified algorithm UCB1 from [2], with a significantly extended analysis. Recall that in each round t algorithm UCB1 chooses an arm u with the highest *index* $I_t(u) = \mu_t(u) + r_t(u)$, where $r_t(u) = \sqrt{8\log(t)/n_t(u)}$ is a term that we'll call the *confidence radius* whose meaning is that $|p(u) - \mu_t(u)| \leq r_t(u)$ with high probability. For our purposes here it is instructive to re-write the index as $I_t(u) = \mu_t(u) + \alpha\, r_t(u)$ for some parameter α. Also, to better bound the early failure probability we will re-define the confidence radius as $r_t(u) = \sqrt{8\log(t_0+t)/n_t(u)}$ for some parameter t_0. We will denote this parameterized version by UCB1(α, t_0).

The original regret analysis of UCB1 in [2] carries over to UCB1(α, t_0) so as to guarantee event-free regret (5); we omit the details.

Our contribution concerns estimating the $\Delta(u)$'s. We estimate the maximal expected reward p^* via the sample average of an arm that has been played most often. More precisely, in order to bound the failure probability we consider an arm that has been played most often *in the last $t/2$ rounds*. For a given round t let v_t be one such arm (ties broken arbitrarily), and let $\Delta_t(u) = \mu_t(v_t) - \mu_t(u)$ will be our estimate of $\Delta(u)$. This estimate (and the provable guarantee thereon) is the main technical contribution of this section.

We obtain an (L,ϵ)-testable algorithm from UCB1$(6, T)$, where T is the time horizon, by defining the t-th round guess as

$$(G^+, G^-) = (\{v : \Delta_t(v) \leq \epsilon/4\},\ \{v : \Delta_t(v) > \epsilon/2\}). \tag{6}$$

The pseudocode is in Algorithm 2.

Let us pass to the provable guarantees. We express the "quality" of our estimate Δ_t as follows:

Theorem 2. *Consider the stochastic n-armed bandits problem. Suppose algorithm* UCB1$(6, t_0)$ *has been played for t steps, and $t + t_0 \geq 32$. Then with probability at least $1-(t_0+t)^{-2}$ for any arm u we have*

$$|\Delta(u) - \Delta_t(u)| < \tfrac{1}{4}\Delta(u) + \delta(t) \tag{7}$$

where $\delta(t) = O(\sqrt{\tfrac{n}{t}\log(t+t_0)})$.

Remark. Either we know that $\Delta(u)$ is small, or we can approximate it up to a constant factor. Specifically, if $\delta(t) < \frac{1}{2}\Delta_t(u)$ then $\Delta(u) \leq 2\,\Delta_t(u) \leq 5\,\Delta(u)$ else $\Delta(u) \leq 4\delta(t)$.

Proof. Fix round t, let $v^* = v_t$ and let s be the last round this arm has been played before round t. Recall that $s \geq t/2$ by definition of v_t. Since by pigeonhole principle $n_t(v^*) \geq \frac{t}{2n}$, it follows that $r_t(v) \leq O(\delta)$ where $\delta = \sqrt{\frac{n}{t}\log(t+t_0)}$. It is easy to see that

$$r_s(v^*) \leq 2\, r_{s+1}(v^*) \leq 2\, r_t(v^*) = O(\delta).$$

Then with probability at least $1-(t_0+t)^{-2}$ for any arm u we have

$$p(v^*) + O(\delta) \geq p(v^*) + 7r_s(v^*) \geq I_s(v^*) \geq I_s(u) \geq p(u) + 5r_s(u). \tag{8}$$

If u^* is the arm with maximal expected reward, then plugging $u = u^*$ into (8) gives $\Delta(v^*) \leq O(\delta)$.

We claim that (8) implies $r_t(u) \le \frac{1}{4}\,\Delta(u) + O(\delta)$. Indeed, we can re-write (8) as

$$5r_s(u) \le p(v^*) - p(u) + O(\delta) \le \Delta(u) + O(\delta).$$

The claim follows since $r_t(u) \le r_s(u)\,\log(t_0+t)/\log(t_0+s) \le \frac{5}{4}\,r_s(u)$.

Now we are ready for the final calculation. Let p^* be the maximal expected reward. Then

$$\begin{aligned}|\Delta(u) - \Delta_t(u)| &= |p^* - p(u) - \mu_t(v^*) + \mu_t(u)| \\ &= |(p^* - p(v^*)) + (p(v^*) - \mu_t(v^*)) + (\mu_t(u) - p(u))| \\ &\le \Delta(v^*) + |p(v^*) - \mu_t(v^*)| + |\mu_t(u) - p(u)| \\ &\le \Delta(v^*) + r_t(v^*) + r_t(u^*) \le \tfrac{1}{4}\,\Delta(u) + O(\delta).\end{aligned}$$ □

Finally, let us prove that Algorithm 2 is (L,ϵ)-testable as long as $L \ge \Omega(\frac{n}{\epsilon^2}\log T)$.

Theorem 3. *Consider algorithm* UCB1$(6,T)$ *where T is the time horizon and the t-th round guess is given by (6). Assume that $\delta(L) \le \epsilon/4$, where $\delta(t)$ is from (7). Then the algorithm is (L,ϵ)-testable.*

Proof. If u is an optimal arm, then $\Delta(u) = 0$, so by (7) we have $\Delta_t(u) \le \delta(t) \le \epsilon/4$. If $\Delta(u) \ge \epsilon$ then by (7) we have $\Delta_t(u) \ge \Delta(u)/2 \ge \epsilon/2$. □

7 Upper and Lower Bounds

Plugging the classifier from Section 5 and the bandit algorithm from Section 6 into the meta-algorithm from Section 4, we obtain the following numerical guarantee.

Theorem 4. *Consider an instance $\mathcal{S}$ of the eventful bandit problem with number of rounds T, n arms, k events, minimum shift ϵ_S, minimum suboptimality Δ, and concept class diameter $d_{\mathcal{F}}$. Assume that any two events are at least $2L$ rounds apart, where $L = \Theta(\frac{n}{\epsilon_S^2}\log T)$. Consider the* BWC *algorithm with parameter L and components* `classifier` *and* `bandit` *as presented, respectively, in Section 5 and Section 6. Then the regret of* BWC *is*

$$R_{\texttt{BWC}}(T) \le \left((3k + 2d_{\mathcal{F}})\tfrac{n}{\Delta} + k\tfrac{n}{\epsilon_S^2}\right)(\log T).$$

While the linear dependence on n in this bound may seem large, note that without additional assumptions, regret must be linear in n, since each arm must be pulled at least once. In an actual search engine application, the arms can be restricted to, say, the top ten results that match the query.

We now state two lower bounds about eventful bandit problems. Theorem 5 shows that in order to achieve regret that is logarithmic in the number of rounds, a context-aware algorithm is necessary, assuming there is at least one event. Incidentally, this lowerbound can be easily extended to prove that, in our model, *no* algorithm can achieve logarithmic regret when an event oracle f is not contained in the concept class $\mathcal{F}$.

Theorem 5. *Consider the eventful bandit problem with number of rounds T, two arms, minimum shift ϵ_S and minimum suboptimality Δ, where $\epsilon_S = \Delta = \epsilon$, for an arbitrary $\epsilon \in (0,\frac{1}{2})$. For any context-ignoring bandit algorithm $\mathcal{A}$, there exists a problem instance with a single event such that regret $R_{\mathcal{A}}(T) \ge \Omega(\epsilon\sqrt{T})$.*

Proof. For simplicity, assume that $N = \sqrt{T}$ is an integer. Define problem instances $\mathcal{I}_i$, $0 \le i \le N$ as follows. In each of these instances, the T rounds are partitioned into N phases, each of length N. There are two arms, call them y and z. Set $p_t(y) = \frac{1}{2}$ for all t. For the problem instance $\mathcal{I}_0$, $p_t(z) = \frac{1}{2} - \epsilon$ for all t. For problem instances $\mathcal{I}_i$, $i \ge 1$ set $p_t(z) = \frac{1}{2} - \epsilon$ in all phases $j < i$, and $p_t(z) = \frac{1}{2} + \epsilon$ in all phases $j \ge i$. (Thus, in each instance $\mathcal{I}_i$ there is a single event that occurs in the first round of phase i.)

Now, let q_i be the probability that on problem instance $\mathcal{I}_0$, arm z is chosen by algorithm $\mathcal{A}$ at least once during phase i. If $q_i \ge \frac{1}{2}$ for each phase i, then on the problem instance $\mathcal{I}_0$ each phase i contributes at least $\epsilon/2$ to regret, so the total regret is at least $\epsilon N/2$. Otherwise, $q_i < \frac{1}{2}$ for some i. Since instances $\mathcal{I}_0$ and $\mathcal{I}_i$ coincide on the first $i-1$ phases, algorithm $\mathcal{A}$ behaves the same way on both instances up to the end of phase $i-1$. Moreover, $\mathcal{A}$ behaves the same way on both instances throughout phase i assuming that it never plays arm z during that phase. Therefore with probability $1-q_i$ its regret on instance $\mathcal{I}_i$ due to phase i alone is ϵ per each round in this phase; so the total regret is at least $\epsilon N/2$. □

Theorem 6 proves that in Theorem 4, linear dependence on $k+d_{\mathcal{F}}$ is essentially unavoidable. If we desire a regret bound that has logarithmic dependence on the number of rounds, then a linear dependence on $k+d_{\mathcal{F}}$ is necessary.

Theorem 6. *Consider the eventful bandit problem with number of rounds T and concept class diameter $d_{\mathcal{F}}$. Let $\mathcal{A}$ be an eventful bandit algorithm.*

(i) There exists a problem instance with n arms, k events, minimum shift ϵ_S, minimum suboptimality Δ, where $\epsilon_S = \Delta = \epsilon$, for arbitrary $k \geq 1$, $n \geq 3$, and $\epsilon \in (0, \frac{1}{4})$, such that $R_{\mathcal{A}}(T) \geq \Omega(k\,\frac{n}{\epsilon})\,\log(T/k)$.

(ii) There exists a problem instance with two arms, a single event, minimum shift $\Theta(1)$ and minimum suboptimality $\Theta(1)$ such that regret $R_{\mathcal{A}}(T) \geq \Omega(T^{1/3})$ or $R_{\mathcal{A}}(T) \geq \Omega(d_{\mathcal{F}} \log T)$.

Proof. For part (i), construct the family of problem instances as follows. In each instance, there are k phases of length T/k each. For each phase i, one arm, call it y_i, has payoff $p_t(y_i) = \frac{1}{2}+\epsilon$, and all other arms y have payoff $p_t(y) = \frac{1}{2}-\epsilon$. We have one problem instance for each sequence $\{y_i\}$ such that $y_i \neq y_{i+1}$ for each i. Note that there is an event in the first round of each phase; without loss of generality let us assume that this is known to the algorithm. Then in each phase $i \geq 1$ the algorithm (essentially) needs to solve a fresh instance of the stochastic bandit problem on $n-1$ arms with time horizon T/k and payoffs $\frac{1}{2} \pm \epsilon$, which implies regret $\Omega(\frac{n}{\epsilon})\log(T/k)$ [16, 2]. We omit the easy formal details.

For part (ii), partition the T rounds into $N = \min(d_{\mathcal{F}}, T^{1/3})$ phases, each of length at least $T^{2/3}$. We define problem instances $\mathcal{I}_i, 0 \leq i \leq N$, in a similar way as in Theorem 5. There are two arms, y and z. Set $p_t(y) = \frac{1}{2}$ for all t. For problem instance $\mathcal{I}_0$, set $p_t(z) = \frac{1}{1+e^{1/3}}$. In problem instance $\mathcal{I}_i$, for $i \geq 1$, set $p_t(z) = \frac{1}{1+e^{1/3}}$ in all phases $j < i$, and $p_t(z) = \frac{e^{1/3}}{1+e^{1/3}}$ in all phases $j \geq i$. Note that $\frac{1}{1+e^{1/3}} < \frac{1}{2} < \frac{e^{1/3}}{1+e^{1/3}}$.

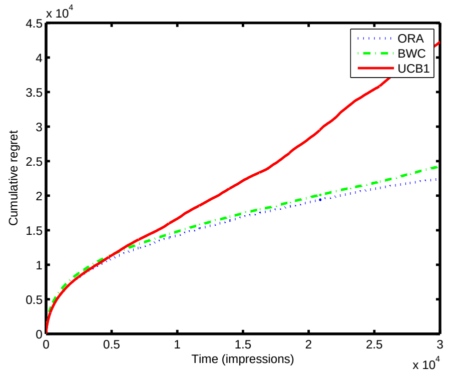
In Appendix A, we show how to define the context sequence $\{x_t\}$ in a way consistent with all our assumptions, in such a way that the contexts for problem instances $\mathcal{I}_0$ and $\mathcal{I}_i$ agree in the first i phases. The idea is that for both problem instances, the first round of each phase $j < i$ triggers a false positive; this is possible since (essentially) we are allowed $d_{\mathcal{F}}$ false positives.

The rest of the proof involves calculations similar to those in the proof of Theorem 5. First, suppose $d_{\mathcal{F}} \geq T^{1/3}$. Define q_i as in the proof of Theorem 5. If $q_i \geq \frac{1}{2}$ for each phase j, then for the problem instance $\mathcal{I}_0$ we have $R_{\mathcal{A}}(T) \geq \Omega(T^{1/3})$. Otherwise, let i be such that $q_i < \frac{1}{2}$. By our construction, with probability $1-q_i$ algorithm $\mathcal{A}$ behaves identically on instances $\mathcal{I}_0$ and $\mathcal{I}_i$ through the first i phases. Thus, on instance $\mathcal{I}_i$ in phase i alone it incurs regret $\Omega(1)$ per each round of the phase, for a total of $R_{\mathcal{A}}(T) \geq \Omega(T^{2/3})$.

Next, suppose $d_{\mathcal{F}} < T^{1/3}$. Let $q_{i,j}$ be the probability that for problem instance $\mathcal{I}_j$, arm z is chosen by $\mathcal{A}$ at least $\log T$ times during phase i. If $q_{i,0} \geq \frac{1}{2}$ for each phase i, then $R_{\mathcal{A}}(T) \geq \Omega(d_{\mathcal{F}} \log T)$ on problem instance $\mathcal{I}_0$. Otherwise, let i be such that $q_{i,0} < \frac{1}{2}$. In Appendix A, we give a calculation that shows that $(1-q_{i,i}) \geq T^{-1/3}(1-q_{i,0})$, which implies that $R_{\mathcal{A}}(T) \geq \Omega(\frac{1}{T^{1/3}})(T^{2/3}-\log T) \geq \Omega(T^{1/3})$ on problem instance $\mathcal{I}_i$. □

8 Experiments

To truly demonstrate the benefits of BWC requires real-time manipulation of search results. Since we did not have the means to deploy a system that monitors click/skip activity and correspondingly alters search results with live users, we describe a collection of experiments on synthetically generated data.



	0	1/8	1/4	3/8	1/2
ORA	17.2	22.8	30.4	33.8	39.5
BWC	17.8	24.6	39.9	46.7	99.4
UCB1	17.2	34.1	114.9	84.2	140.0
EXP3.S	78.4	123.7	180.2	197.6	243.1

	10	20	30	40
ORA	21.9	23.2	21.9	22.8
BWC	23.1	24.4	22.9	23.7
UCB1	32.3	33.5	31.1	37.4
EXP3.S	111.6	109.4	112.5	121.3

Figure 2: (a) (Left) BWC's cumulative regret compared to UCB1 and ORA (UCB1 with an oracle indicating the exact locations of the intent-shifting event) (b) (Right, Top Table) Final regret (in thousands) as the fraction of intent-shifting queries varies. With more intent-shifting queries, BWC's advantage over prior approaches improves. (c) (Right, Bottom Table) Final regret (in thousands) as the number of features grows.

We begin with a head-to-head comparison of BWC versus a baseline UCB1 algorithm and show that BWC's performance improves substantially upon UCB1. Next, we compare the performance of these algorithms as we vary the fraction of intent-shifting queries: as the fraction increases, BWC's performance improves even further upon prior approaches. Finally, we compare the performance as we vary the number of features. While our theoretical results suggest that regret grows with the number of features in the context space, in our experiments, we surprisingly find that BWC is robust to higher dimensional feature spaces.

Setup: We synthetically generate data as follows. We assume that there are 100 queries where the total number of times these queries are posed is 3M. Each query has five search results for a user to select from. If a query does not experience any events — i.e., it is not "intent-shifting" — then the optimal search result is fixed over time; otherwise the optimal search result may change. Only 10% of the queries are intent-shifting, with at most 10 events per such query. Due to the random nature with which data is generated, regret is reported as an average over 10 runs. The event oracle is an axis-parallel rectangle anchored at the origin, where points inside the box are negative and points outside the box are positive. Thus, if there are two features, say query volume and query abandonment rate, an event occurs if and only if both the volume and abandonment rate exceed certain thresholds.

Bandit with Classifier (BWC): Figure 2(a) shows the average cumulative regret over time of three algorithms. Our baseline comparison is UCB1 which assumes that the best search result is fixed throughout. In addition, we compare to an algorithm we call ORA, which uses the event oracle to reset UCB1 whenever an event occurs. We also compared to EXP3.S, but its performance was dramatically worse and thus we have not included it in the figure.

In the early stages of the experiment before any intent-shifting event has happened, UCB1 performs the best. BWC's safe classifier makes many mistakes in the beginning and consequently pays the price of believing that each query is experiencing an event when in fact it is not. As time progresses, BWC's classifier makes fewer mistakes, and consequently knows when to reset UCB1 more accurately. UCB1 alone ignores the context entirely and thus incurs substantially larger cumulative regret by the end.

Fraction of Intent-Shifting Queries: In the next experiment, we varied the fraction of intent-shifting queries. Figure 2(b) shows the result of changing the distribution from 0, 1/8, 1/4, 3/8 and 1/2 intent-shifting queries. If there are no intent-shifting queries, then UCB1's regret is the best. We expect this outcome since BWC's classifier, because it is safe, initially assumes that all queries are intent-shifting and thus needs time to learn that in fact no queries are intent-shifting. On the other hand, BWC's regret dominates the other approaches, especially as the fraction of intent-shifting queries grows. EXP3.S's performance is quite poor in this experiment — even when all queries are intent-shifting. The reason is that even when a query is intent-shifting, there are at most 10 intent-shifting events, i.e., each query's intent is not shifting all the time.

With more intent-shifting queries, the expectation is that regret monotonically increases. In general, this seems to be true in our experiment. There is however a decrease in regret going from 1/4 to 3/8 intent-shifting queries. We believe that this is due to the fact that each query has at most 10 intent-shifting events spread uniformly and it is possible that there were fewer events with potentially smaller shifts in intent in those runs. In other words, the standard deviation of the regret is large. Over the ten 3/8 intent-shifting runs for ORA, BWC, UCB1 and EXP3.S, the standard deviation was roughly 1K, 10K, 12K and 6K respectively.

Number of Features: Finally, we comment on the performance of our approach as the number of features grows. Our theoretical results suggest that BWC's performance should deteriorate as the number of features grows. Surprisingly, BWC's performance is consistently close to the Oracle's. In Figure 2(b), we show the cumulative regret after 3M impressions as the dimensionality of the context vector grows from 10 to 40 features. BWC's regret is consistently close to ORA as the number of features grows. On the other hand, UCB1's regret though competitive is worse than BWC, while EXP3.S's performance is across the board poor. Note that both UCB1 and EXP3.S's regret is completely independent of the number of features. The standard deviation of the regret over the 10 runs is substantially lower than the previous experiment. For example, over 10 features, the standard deviation was 355, 1K, 5K, 4K for ORA, BWC, UCB1 and EXP3.S, respectively.

9 Future Work

The most immediate open question is whether we could train the classifier faster. One idea is to use a more efficient classifier, especially if we can relax the "safety" requirement and somehow recover from false negatives. Another idea is to generate labeled samples not only upon positive predictions but upon negative ones as well, trading off the regret from additional exploration against the benefits of generating extra labeled samples. Finally, it would be desirable to supplement the existing worst-case provable guarantees with stronger ones for settings in which the contexts are sampled from a "benign" distribution.

Theoretically, the main drawback of our approach is that we assume the existence of a "perfect oracle" — a deterministic boolean function on contexts which correctly predicts whether a temporal event has occurred in the current round. It is desirable to extend our results to scenarios in which the contexts allow only approximate or probabilistic prediction. Even though such contexts contain useful signal, exploiting this signal for our purposes appears quite challenging. In particular, it seems to require making the "bandit plus classifier" setup resilient against (infrequent) incorrectly labeled samples and perhaps also against (infrequent) false negatives. It should be noted that the aforementioned resiliency can potentially lead to large improvements in the present oracle-based setting as well, as we might be able to deploy much more efficient classifiers.

Empirically, the main question left for future work is testing the "bandit plus classifier" approach in a realistic setting. The challenge here is two-fold. First, one needs to select which features to use for contexts, and verify experimentally how informative they are in predicting the temporal events. Second, since gaining access to live search traffic is difficult, one would need to simulate it using the search logs, the difficulty being is that the search logs might not have enough data points for alternatives that have not been chosen frequently by the search engine.

Acknowledgements. We thank Rakesh Agrawal, Alan Halverson, Krishnaram Kenthapadi, Robert Kleinberg, Robert Schapire and Yogi Sharma for their helpful comments and suggestions.

References

[1] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, Nitin Motgi, Seung-Taek Park, Raghu Ramakrishnan, Scott Roy, and Joe Zachariah. Online models for content optimization. In *22nd Advances in Neural Information Processing Systems (NIPS)*, 2008.

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[3] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

[4] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[5] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *22nd Intl. Conf. on Machine Learning (ICML)*, 2005.

[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *24th Intl. Conf. on Machine Learning (ICML)*, 2007.

[7] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[8] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *J. of Artificial Intelligence Research*, 10:243–270, 1999.

[9] Fernando Diaz. Integration of news content into web results. In *2nd Intl. Conf. on Web Search and Data Mining*, pages 182–191, 2009.

[10] D. Fallows. Search engine users. *Pew Internet and American Life Project*, 2005.

[11] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. of Machine Learning Research*, 4:933–969, 2003.

[12] Elad Hazan and Nimrod Megiddo. Online Learning with Prior Knowledge. In *20th Conference on Learning Theory (COLT)*, pages 499–513, 2007.

[13] Thorsten Joachims. Optimizing search engines using clickthrough data. In *8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2002.

[14] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *25th Intl. Conf. on Machine Learning (ICML)*, 2008.

[15] Jon M. Kleinberg. Bursty and hierarchical structure in streams. In *8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2002.

[16] T.L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[17] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *21st Advances in Neural Information Processing Systems (NIPS)*, 2007.

[18] Sandeep Pandey, Deepak Agarwal, Deepayan Chakrabarti, and Vanja Josifovski. Bandits for Taxonomies: A Model-based Approach. In *SIAM Intl. Conf. on Data Mining (SDM)*, 2007.

[19] Sandeep Pandey, Deepayan Chakrabarti, and Deepak Agarwal. Multi-armed Bandit Problems with Dependent Arms. In *24th Intl. Conf. on Machine Learning (ICML)*, 2007.

[20] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *25th Intl. Conf. on Machine Learning (ICML)*, 2008.

[21] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

[22] Chih-Chun Wang, Sanjeev R. Kulkarni, and H. Vincent Poor. Bandit problems with side observations. *IEEE Trans. on Automatic Control*, 50(3):338355, 2005.

[23] Jia Yuan Yu and Shie Mannor. Piecewise-stationary bandit problems with side observations. In *26th Intl. Conf. on Machine Learning (ICML)*, 2009.

A Details for the proof of Theorem 6(ii)

Claim 5. *We can define context sequences $\{x^0_t\}$ and $\{x^1_t\}, \ldots, \{x^N_t\}$ with the following properties: (1) each sequence $\{x^i_t\}$, when paired with a problem instance $\mathcal{I}_i$, defines an eventful bandit problem consistent with all our assumptions, and (2) the sequences $\{x^0_t\}$ and $\{x^i_i\}$ agree through the first i phases.*

Proof. Let $y_1, \ldots, y_{d_\mathcal{F}} \in \mathcal{X}$ be a sequence of contexts such that $y_j \notin S_\mathcal{F}(\{y_1, \ldots, y_{j-1}\})$ for all $j = 1, \ldots, d_\mathcal{F}$. We know this sequence exists by the definition of $d_\mathcal{F}$. Also assume there exists an "always negative" context x^- such that $f(x^-) = -1$ for all $f \in \mathcal{F}$ (this assumption is not necessary, but is convenient). Let t_j be the first round of phase j.

Define $\{x^0_t\}$ as follows: let $x^0_{t_j} = y_j$ for each phase $1 \leq j \leq N$, and let $x^0_t = x^-$ for all other rounds.

For $1 \leq i \leq N$, define $\{x^i_t\}$ as follows: let $x^i_{t_j} = y_j$ for each phase $1 \leq j \leq i$, and let $x^i_t = x^-$ for all other rounds. ☐

Claim 6. $(1 - q_{i,i}) \geq T^{-1/3}(1 - q_{i,0})$

Proof. Throughout this proof, we fix phase i. Define a *realization* to be a particular sequence of outcomes of all random samples from click distributions, as well as all random choices (if any), during an execution of algorithm A *through the end of phase* i. For example, if $s = s_1, \ldots, s_M$ is a realization, then s_1 might correspond to the click observed in the first round, $s_2, \ldots, s_5$ might correspond to random choices made by the algorithm, s_6 might correspond to the click observed in the second round, and so on. By the chain rule, for any $j \in \{0, i\}$:

$$\Pr_{\mathcal{I}_j}[s] = \Pr_{\mathcal{I}_j}[s_1] \Pr_{\mathcal{I}_j}[s_2|s_1] \cdots \Pr_{\mathcal{I}_j}[s_M|s_1, \ldots, s_{M-1}]$$

For any realization s, let $\Pr_{\mathcal{I}_j}[s_\alpha]$ be the product of terms in the above product that correspond to outcomes other than observed clicks in phase i. Let $\mathcal{S}$ be the set of realizations in which arm z is selected by $\mathcal{A}$ less than $\log T$ times in phase i. Let $n_{a,c}(s)$ be the number of times in realization s that arm $a \in \{y, z\}$ is selected in phase i and payoff $c \in \{0, 1\}$ is observed as a result. Then

$$\begin{aligned}
(1 - q_{i,0}) &= \sum_{s \in \mathcal{S}} \Pr_{\mathcal{I}_0}[s] \\
&= \sum_{s \in \mathcal{S}} \Pr_{\mathcal{I}_0}[s_\alpha] \left(\tfrac{1}{2}\right)^{n_{y,0}(s)} \left(\tfrac{1}{2}\right)^{n_{y,1}(s)} \left(\tfrac{e^{1/3}}{1+e^{1/3}}\right)^{n_{z,0}(s)} \left(\tfrac{1}{1+e^{1/3}}\right)^{n_{z,1}(s)} \\
&= \sum_{s \in \mathcal{S}} \Pr_{\mathcal{I}_0}[s_\alpha] \left(\tfrac{1}{2}\right)^{n_{y,0}(s)} \left(\tfrac{1}{2}\right)^{n_{y,1}(s)} \left(e^{1/3} \cdot \tfrac{1}{1+e^{1/3}}\right)^{n_{z,0}(s)} \left(\tfrac{1}{e^{1/3}} \cdot \tfrac{e^{1/3}}{1+e^{1/3}}\right)^{n_{z,1}(s)} \\
&\leq \sum_{s \in \mathcal{S}} \Pr_{\mathcal{I}_0}[s_\alpha] (e^{1/3})^{n_{z,0}(s)} \left(\tfrac{1}{2}\right)^{n_{y,0}(s)} \left(\tfrac{1}{2}\right)^{n_{y,1}(s)} \left(\tfrac{1}{1+e^{1/3}}\right)^{n_{z,0}(s)} \left(\tfrac{e^{1/3}}{1+e^{1/3}}\right)^{n_{z,1}(s)} \\
&\leq (e^{1/3})^{\log T} \sum_{s \in \mathcal{S}} \Pr_{\mathcal{I}_0}[s_\alpha] \left(\tfrac{1}{2}\right)^{n_{y,0}(s)} \left(\tfrac{1}{2}\right)^{n_{y,1}(s)} \left(\tfrac{1}{1+e^{1/3}}\right)^{n_{z,0}(s)} \left(\tfrac{e^{1/3}}{1+e^{1/3}}\right)^{n_{z,1}(s)} \\
&= T^{1/3} \sum_{s \in \mathcal{S}} \Pr_{\mathcal{I}_i}[s] \\
&= T^{1/3} (1 - q_{i,i})
\end{aligned}$$

□